题目要求：
利用PLY实现的Python程序的解析

本次学习的语法是函数语句，需要注意的是本次使用的语法做了一些改进，不是纯粹的python2语法。

需要结合上次课四则运算的解析程序

（1）示例程序位于example4/

（2）需要进行解析的文件为快速排序quick_sort.py

（3）解析结果以语法树的形式呈现

quick_sort.py文件内容如下：

```
def quick_sort(array, left, right):
    if(left >= right){
        return
    }
    low = left
    high = right
    key = array[low]
    while(left < right){
        while(left < right and array[right] > key){
            right -= 1
        }
        array[left] = array[right]
        while(left < right and array[left] <= key){
            left += 1
        }
        array[right] = array[left]
    }
    array[right] = key
    quick_sort(array, low, left - 1)
    quick_sort(array, left + 1, high)
a=[1,2,4,3,6,5,7,3]
quick_sort(a,0,len(a)-1)
print(a)
```

程序说明：

1. 打开main.py文件，确保source中的所有代码在同一目录下

2. 确保已经安装了PLY库

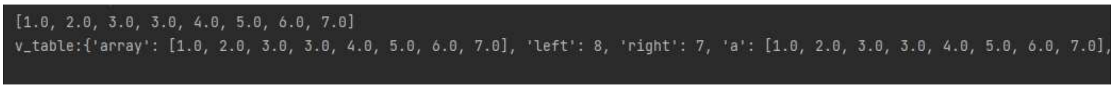
3. 运行 main.py 文件

4. 对 quick_sort.py 文件中的程序段进行解析，结果以语法树的形式展现，并展示 print 的结果以及所有变量的最终值字典，解析结果如下(图片太长，直接放文字)：

```
+ [PROGRAM]
  + [STATEMENTS]
    + [STATEMENTS]
      + [STATEMENTS]
        + [STATEMENTS]
          + [STATEMENTS]
            + [STATEMENT]
              + [FUNCTION]
                + quick_sort
                + [SENTENCE]
                  + [WORD]
                    + array
                  + ,
                  + [SENTENCE]
                    + [WORD]
                      + left
                    + ,
                    + [SENTENCE]
                      + [WORD]
                        + right
                + [STATEMENTS]
                  + [STATEMENTS]
                    + [STATEMENTS]
                      + [STATEMENTS]
                        + [STATEMENTS]
                          + [STATEMENTS]
                            + [STATEMENTS]
                              + [STATEMENTS]
                                + [STATEMENTS]
                                  + [STATEMENTS]
                                    + [STATEMENT]
                                      + [IF]
                                        + [CONDITION]
                                          + left
                                          + >
                                          + =
                            + right
                        + [STATEMENTS]
                            + [STATEMENT]
                                + [RETURN]
                                    + return
            + [STATEMENT]
                + [OPERATION]
                    + low
                    + =
                    + [EXPR]
                        + [TERM]
                            + [FACTOR]
                                + left
        + [STATEMENT]
            + [OPERATION]
                + high
                + =
                + [EXPR]
                    + [TERM]
                        + [FACTOR]
                            + right
    + [STATEMENT]
        + [OPERATION]
            + key
            + =
            + [EXPR]
                + array
                + [
                + [FACTOR]
                    + low
                + ]
+ [STATEMENT]
    + [WHILE]
        + [CONDITION]
            + left
            + <
            + right
        + [STATEMENTS]
            + [STATEMENTS]
                + [STATEMENTS]
                    + [STATEMENTS]
                        + [STATEMENT]
                            + [WHILE]
                                +
[CONDITION_COMPLEX1]
                                              + left
                                              + right
                                              + array
                                              + [FACTOR]
                                                + right
                                              + key
                                            + [STATEMENTS]
                                              + [STATEMENT]
                                                + [OPERATION]
                                                  + right
                                                  + -
                                      + [STATEMENT]
                                        + [MODIFICATION]
                                          + array
                                          + [FACTOR]
                                            + left
                                          + array
                                          + [FACTOR]
                                            + right
                                    + [STATEMENT]
                                      + [WHILE]
                                        + [CONDITION_COMPLEX2]
                                          + left
                                          + right
                                          + array
                                          + [FACTOR]
                                            + left
                                          + key
                                        + [STATEMENTS]
                                          + [STATEMENT]
                                            + [OPERATION]
                                              + left
                                              + +
                                  + [STATEMENT]
                                    + [MODIFICATION]
                                      + array
                                      + [FACTOR]
                                        + right
                                      + array
                                      + [FACTOR]
                                        + left
                          + [STATEMENT]
                            + [MODIFICATION]
```

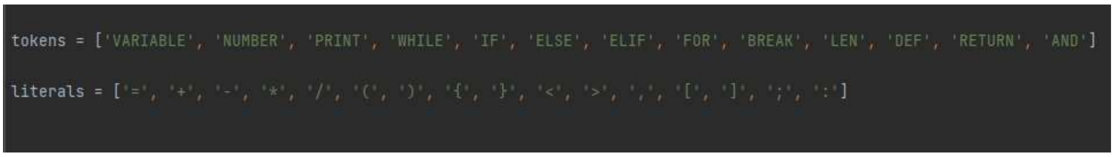
```
               + array
               + [FACTOR]
                  + right
               + key
      + [STATEMENT]
         + [OPERATION]
            + x
            + =
            + [EXPR]
               + [EXPR]
                  + [TERM]
                     + [FACTOR]
                        + left
               + -
               + [TERM]
                  + [FACTOR]
                     + 1
   + [STATEMENT]
      + [OPERATION]
         + y
         + =
         + [EXPR]
            + [EXPR]
               + [TERM]
                  + [FACTOR]
                     + left
            + +
            + [TERM]
               + [FACTOR]
                  + 1
+ [STATEMENT]
   + [RUNFUNCTION]
      + quick_sort
      + [SENTENCE]
         + [WORD]
            + array
         + ,
         + [SENTENCE]
            + [WORD]
               + low
            + ,
            + [SENTENCE]
               + [WORD]
                  + x
            + [STATEMENT]
               + [RUNFUNCTION]
                  + quick_sort
                  + [SENTENCE]
                     + [WORD]
                        + array
                     + ,
                     + [SENTENCE]
                        + [WORD]
                           + y
                        + ,
                        + [SENTENCE]
                           + [WORD]
                              + high
+ [STATEMENT]
   + [ASSIGNMENT]
      + a
      + =
      + [
      + [SENTENCE]
         + [WORD]
            + 1
         + ,
         + [SENTENCE]
            + [WORD]
               + 2
            + ,
            + [SENTENCE]
               + [WORD]
                  + 4
               + ,
               + [SENTENCE]
                  + [WORD]
                     + 3
                  + ,
                  + [SENTENCE]
                     + [WORD]
                        + 6
                     + ,
                     + [SENTENCE]
                        + [WORD]
                           + 5
                        + ,
                        + [SENTENCE]
                        + [WORD]
                          + 3
                        + ,
                        + [SENTENCE]
                          + [WORD]
                            + 7
          + ]
    + [STATEMENT]
      + [OPERATION]
        + b
        + =
        + [EXPR]
          + [EXPR]
            + len(
            + [TERM]
              + [FACTOR]
                + a
            + )
          + -
          + [TERM]
            + [FACTOR]
              + 1
  + [STATEMENT]
    + [RUNFUNCTION]
      + quick_sort
      + [SENTENCE]
        + [WORD]
          + a
        + ,
        + [SENTENCE]
          + [WORD]
            + 0
          + ,
          + [SENTENCE]
            + [WORD]
              + b
+ [STATEMENT]
  + [PRINT]
    + print
    + (
    + [SENTENCE]
      + [WORD]
        + a
    + )
```

程序的 print 结果以及对应的最后 v_table 内容如下:



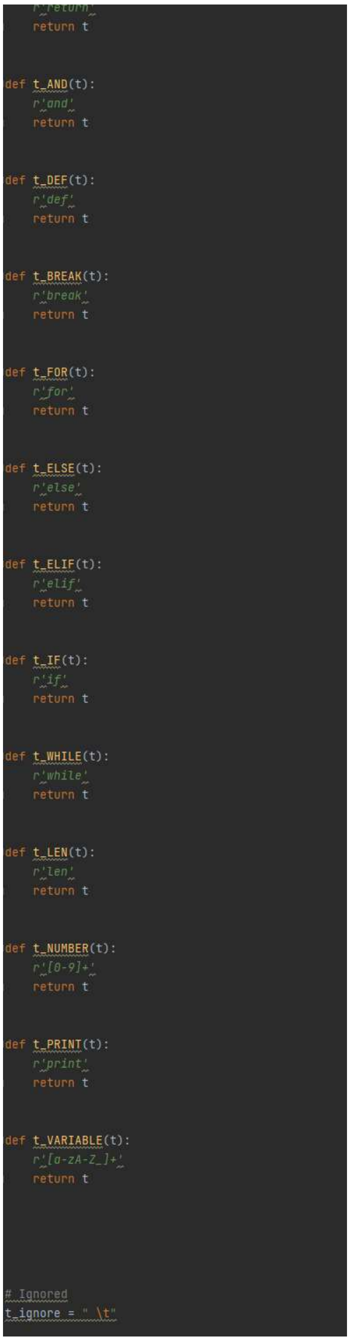
```
[1.0, 2.0, 3.0, 3.0, 4.0, 5.0, 6.0, 7.0]
v_table:{'array': [1.0, 2.0, 3.0, 3.0, 4.0, 5.0, 6.0, 7.0], 'left': 8, 'right': 7, 'a': [1.0, 2.0, 3.0, 3.0, 4.0, 5.0, 6.0, 7.0],
```

5. 对 Lexer 程序定义的 token 规则的解释



```
tokens = ['VARIABLE', 'NUMBER', 'PRINT', 'WHILE', 'IF', 'ELSE', 'ELIF', 'FOR', 'BREAK', 'LEN', 'DEF', 'RETURN', 'AND']

literals = ['=', '+', '-', '*', '/', '(', ')', '{', '}', '<', '>', ',', '[', ']', ';', ':']
```



```
    r'return'
    return t


def t_AND(t):
    r'and'
    return t


def t_DEF(t):
    r'def'
    return t


def t_BREAK(t):
    r'break'
    return t


def t_FOR(t):
    r'for'
    return t


def t_ELSE(t):
    r'else'
    return t


def t_ELIF(t):
    r'elif'
    return t


def t_IF(t):
    r'if'
    return t


def t_WHILE(t):
    r'while'
    return t


def t_LEN(t):
    r'len'
    return t


def t_NUMBER(t):
    r'[0-9]+'
    return t


def t_PRINT(t):
    r'print'
    return t


def t_VARIABLE(t):
    r'[a-zA-Z_]+'
    return t




# Ignored
t_ignore = " \t"
```

不难发现，这次的 token 里面多了一些新的关键字，比如说 and，return，def。其余和上次的实验保持一致。值得注意的是这些新加入的关键字的优先级都是更高的，要写在变量那些关键字的上面。

6. Yacc 语法规则的设计

设计的语法规则展开后如下所示：

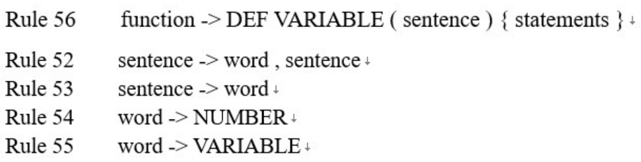
```
 Grammar

Rule 0      S' -> program
Rule 1      program -> statements
Rule 2      statements -> statements statement
Rule 3      statements -> statement
Rule 4      statement -> assignment
Rule 5      statement -> operation
Rule 6      statement -> print
Rule 7      statement -> modification
Rule 8      statement -> iF
Rule 9      statement -> whilE
Rule 10     statement -> for
Rule 11     statement -> break
Rule 12     statement -> return
Rule 13     statement -> function
Rule 14     statement -> runfunction
Rule 15     break -> BREAK statements
Rule 16     break -> BREAK
Rule 17     return -> RETURN
Rule 18     for -> FOR ( operation ; condition ; operation ) { statements }
Rule 19     condition -> VARIABLE > VARIABLE
Rule 20     condition -> VARIABLE < VARIABLE
Rule 21     condition -> VARIABLE > NUMBER
Rule 22     condition -> VARIABLE < NUMBER
Rule 23     condition -> VARIABLE < = VARIABLE
Rule 24     condition -> VARIABLE > = VARIABLE
Rule 25     condition -> VARIABLE [ factor ] > VARIABLE
Rule 26     condition -> VARIABLE [ factor ] < VARIABLE
Rule 27     condition -> VARIABLE < VARIABLE AND VARIABLE [ factor ] >
VARIABLE
Rule 28     condition -> VARIABLE < VARIABLE AND VARIABLE [ factor ] < =
VARIABLE
```

Rule 29 iF -> IF (condition) { statements }
Rule 30 iF -> IF (condition) { statements } ELIF (condition) { statements } ELSE { statements }
Rule 31 whilE -> WHILE (condition) { statements }
Rule 32 assignment -> VARIABLE = NUMBER
Rule 33 assignment -> VARIABLE = [sentence]
Rule 34 modification -> VARIABLE [factor] = VARIABLE [factor]
Rule 35 modification -> VARIABLE [factor] = VARIABLE
Rule 36 operation -> VARIABLE = expression
Rule 37 operation -> VARIABLE + +
Rule 38 operation -> VARIABLE - -
Rule 39 expression -> expression + term
Rule 40 expression -> expression - term
Rule 41 expression -> term
Rule 42 expression -> VARIABLE [factor]
Rule 43 expression -> LEN (term)
Rule 44 term -> term * factor
Rule 45 term -> term / factor
Rule 46 term -> term / / factor
Rule 47 term -> factor
Rule 48 factor -> VARIABLE
Rule 49 factor -> (expression)
Rule 50 factor -> NUMBER
Rule 51 print -> PRINT (sentence)
Rule 52 sentence -> word , sentence
Rule 53 sentence -> word
Rule 54 word -> NUMBER
Rule 55 word -> VARIABLE
Rule 56 function -> DEF VARIABLE (sentence) { statements }
Rule 57 runfunction -> VARIABLE (sentence)

7. Translation 的关键部分逻辑设计

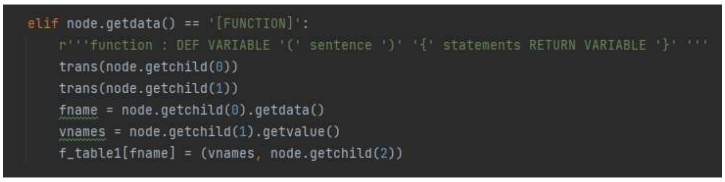
（1） 函数 FUNCTION



Rule 56 function -> DEF VARIABLE (sentence) { statements } ↓

Rule 52 sentence -> word , sentence ↓
Rule 53 sentence -> word ↓
Rule 54 word -> NUMBER ↓
Rule 55 word -> VARIABLE ↓

规则在这，可以发现 sentence 最终可以归约到多个变量上去。



```python
elif node.getdata() == '[FUNCTION]':
    r'''function : DEF VARIABLE '(' sentence ')' '{' statements RETURN VARIABLE '}' '''
    trans(node.getchild(0))
    trans(node.getchild(1))
    fname = node.getchild(0).getdata()
    vnames = node.getchild(1).getvalue()
    f_table1[fname] = (vnames, node.getchild(2))
```

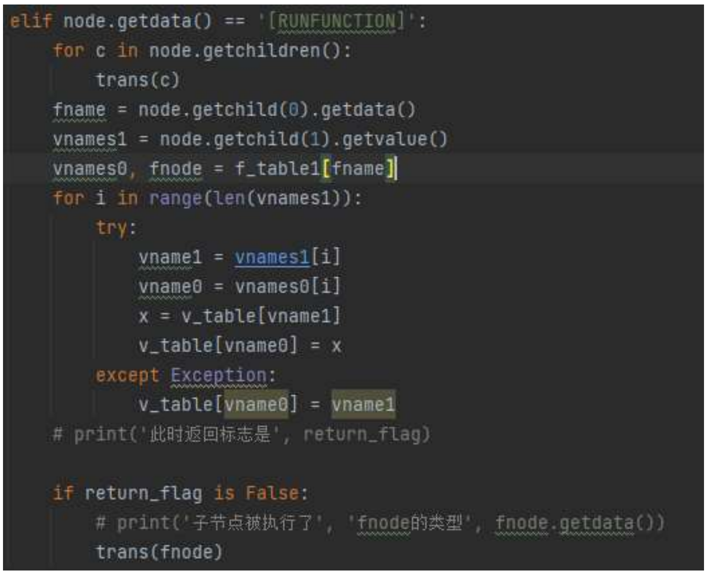
先翻译下头两个结点，也就是变量名和形参 sentence。再获取函数名和自变量数组（这个数组具体的值传递方式在 sentence 部分规定好了），把内容存到放函数的 table 去即可。

（2） 执行函数 RUNFUNCTION

规则如下

Rule 57　　runfunction -> VARIABLE (sentence)

对应到 translation 的代码：



```python
elif node.getdata() == '[RUNFUNCTION]':
    for c in node.getchildren():
        trans(c)
    fname = node.getchild(0).getdata()
    vnames1 = node.getchild(1).getvalue()
    vnames0, fnode = f_table1[fname]
    for i in range(len(vnames1)):
        try:
            vname1 = vnames1[i]
            vname0 = vnames0[i]
            x = v_table[vname1]
            v_table[vname0] = x
        except Exception:
            v_table[vname0] = vname1
    # print('此时返回标志是', return_flag)

    if return_flag is False:
        # print('子节点被执行了', 'fnode的类型', fnode.getdata())
        trans(fnode)
```

先把函数名取出来，然后找到输入函数的实参名数组，从 f_table 里面取出形参数组和待执行的子结点，把实参赋值给形参，如果此时函数并没有执行返回语句，那么就 Translate 子结点。

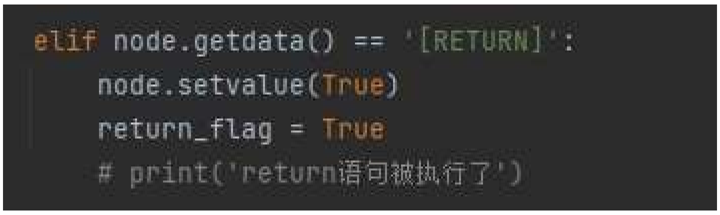
（3） 关于 return 的信号怎么层层传递回 statements 去，让函数停止运行。

Return 的规则：

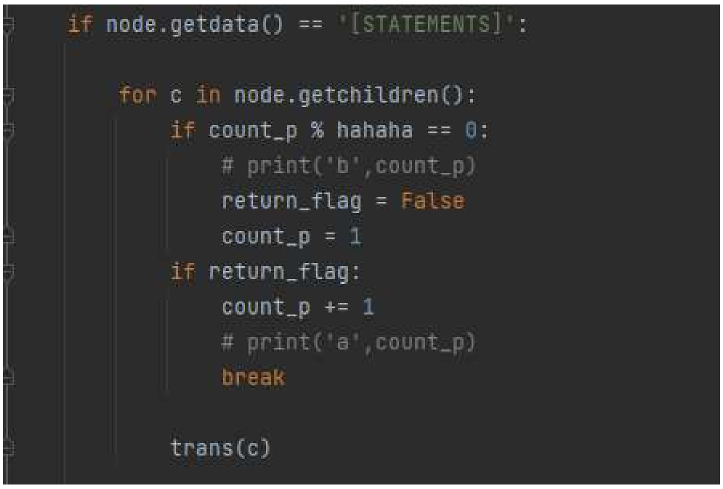
Rule 12　　statement -> return

首先设置一个 return_flag 用于表示此时是否 return 了，如果 return 了那么该值为 True，否则为 False。然后设置一个变量用于统计从函数执行的那一层层层返回到应该继续往下执行的那一层 statements 中间一共几层（自底向上），这个工作在 statements 自顶向下递归的时候统计完成。然后就不停地向上 break 掉循环，直到计数器 count 可以整除该变量，那么就停止 break，正常循环，重置 count，并把 return_flag 重新设置为 False。

代码：



```
elif node.getdata() == '[RETURN]':
    node.setvalue(True)
    return_flag = True
    # print('return语句被执行了')
```



```
if node.getdata() == '[STATEMENTS]':

    for c in node.getchildren():
        if count_p % hahaha == 0:
            # print('b',count_p)
            return_flag = False
            count_p = 1
        if return_flag:
            count_p += 1
            # print('a',count_p)
            break

        trans(c)
```

（其中 hahaha 即为统计向下递归层数的变量）